DAY 2: HACKATHON 3

“MARKETPLACE TECHNICAL FOUNDATION-NIKE STORE”

A. Technical requirement:

1. Frontend Requirement:

- UI/UX design and website interface should be user friendly because people don't understand the complicated website and it's not easy to use. It should be handy.
- Website or app should be responsive according to the screen size, since most users access websites via their smartphones.
- The website should have a home/ landing page which has basic information about the platform.
- Login or sign up functionality is mandatory to allow the owner to gather customer information and their background.
- Product listing, product detail, cart and checkout page is also essential as it has the basic information related to products which are present in the store

2. Sanity CMS as backend:

- The website, which I'm going to launch is General e-commerce, which has clothing, shoes for men women and kids. Sanity CMS is to be used as a backend because customers visit the website to buy products. Sanity CMS is used to manage data of products and customers and it also acts as a database for my website.
- As soon as the customer logged into the website the CMS gets all the data and stores it like the customer ID, customer name, email, phone number, address. If the customer places an order then the sanity CMS (backend) generates an order by fetching some information given by the customer, which means which product he wants to buy, quantity of the product etc.
- This sanity now generates orders and processes shipment by taking information related to shipment. Example: tracking ID, address, country / city delivery charges delivery date.

3. Third Party APIs:

- Third-party APIs will be integrated into the website for the shipment process. These APIs will allow customers to track their orders, know the expected delivery date, and trace the rider handling the shipment.
- The APIs will also enable the shop's location to be shared with the customers for added convenience.
- Additionally, third-party APIs will be used for social login options, such as Google and Facebook. These platforms have robust databases, allowing customers to log in easily and securely via these integrations.

B. Plan API Requirement:

1. END-POINT: /products

Method: GET	This GET method is used to retrieve information.
Description:	Fetch details of all available products and store them in sanity CMS.
Response Example:	Product details: ID, name, price, stock, image, category.

2. END-POINT: /orders

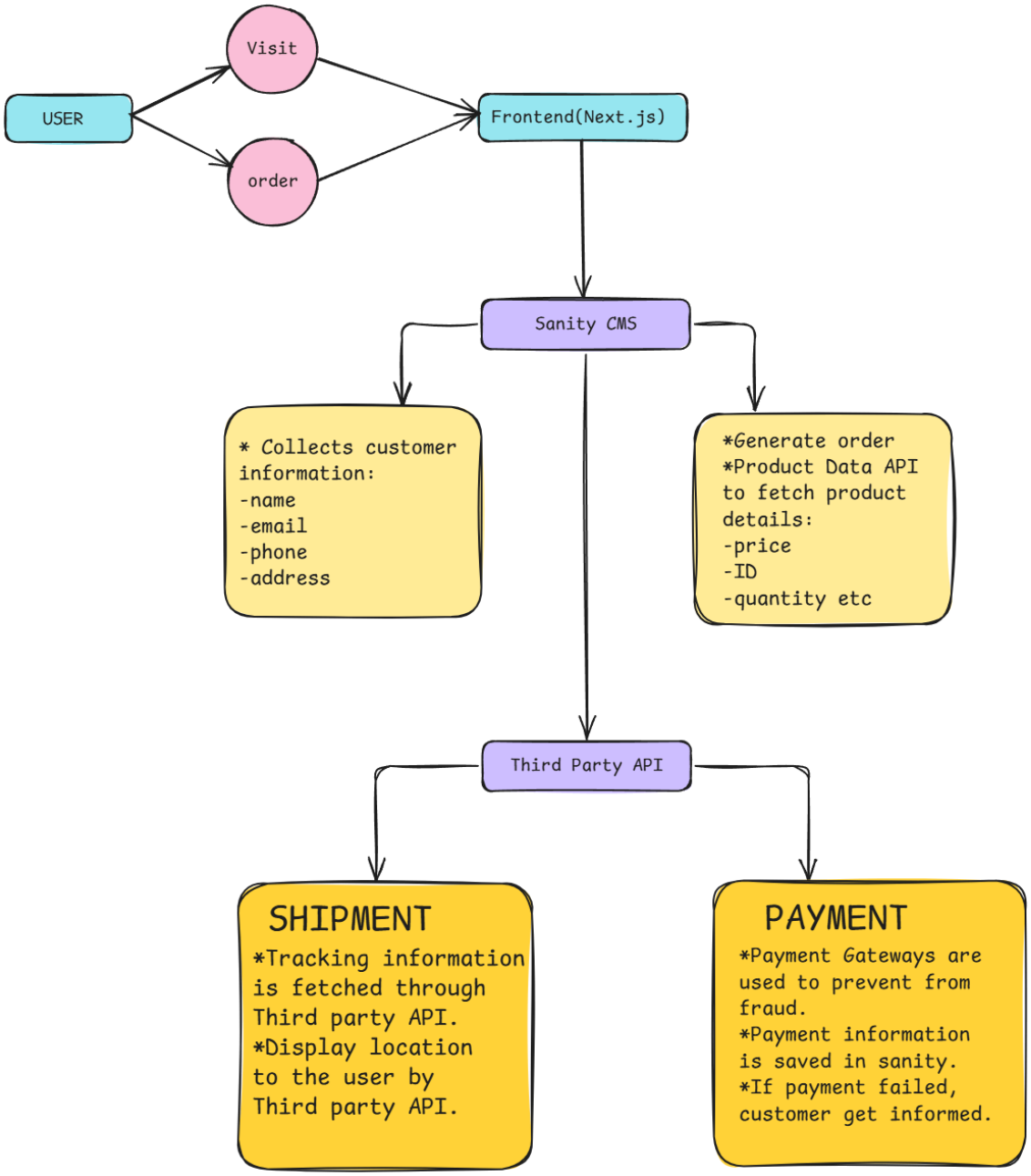
Method: POST	This method is used to create a new order in sanity.
Description:	When a customer places an order, the new order details are stored in sanity.
Payload:	It includes: Customer information: Name, address, contact detail. Product detail: Ordered product ID, price, stock, name, category, image. Payment status: Completed, pending, failed.

3. END-POINT: /shipment

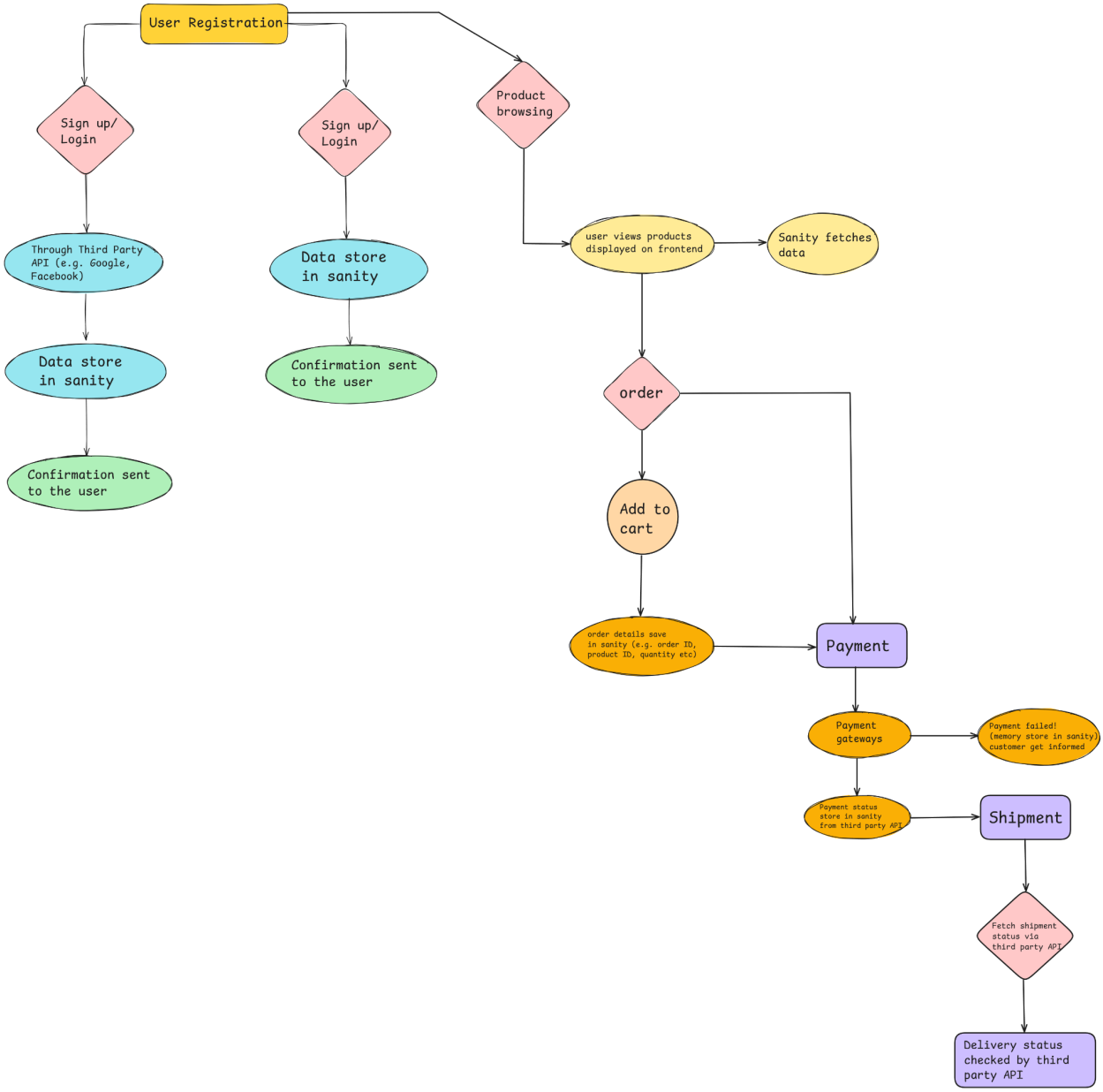
Method: GET	This GET method is used to retrieve shipment status after order.

Description:	Third party API integrates to track order status whether it is pending, failed, delivered.
Response Example:	Shipment ID, shipment status, order ID, delivery date.

C. Technical requirement:



KEY WORKFLOWS TO INCLUDE:



D. Technical documentation:

"Marketplace Technical Foundation - Nike Store"

1. **System architecture overview:**

- Find a store ---> all-Product (page) ---> data from sanity CMS
- Get help ---> contact (page) ---> contact, page location ---> integrates Third party API
- Join us ---> join-us (page) ---> third party API
- Sign in ---> sign-in (page) ---> third party API
- Add to cart button ---> cart page
- Member checkout button ---> checkout page

2. **Category Specific Instruction:**

   user ---> order ---> end-point:/order ---> Method: POST ---> fetch order details and store in sanity ---> fetch payment detail, product detail, customer detail for shipment

3. **Sanity Schema Example:**

```
export default{
name: "product",
type: "document",
title: "Product",
fields: [
{name: "name", type: "string", title: "Product name"},
{name: "price", type: "number", title: "Price"},
{name: "description", type: "string", title: "Description"},
{name: "stock", type: "number", title: "Stock"}
]
```